Generalization in portfolio-based algorithm selection


Nina Balcan, Tuomas Sandholm, **Ellen Vitercik**

Carnegie Mellon University

AAAI'21

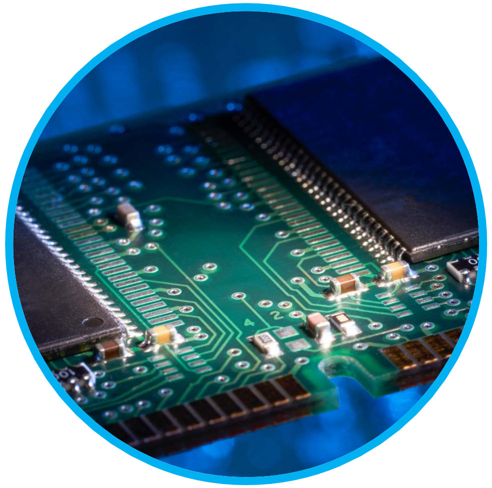
Algorithm parameters

Algorithms often have **many tunable parameters**

Significant impact on:

Runtime

Solution quality



Memory usage

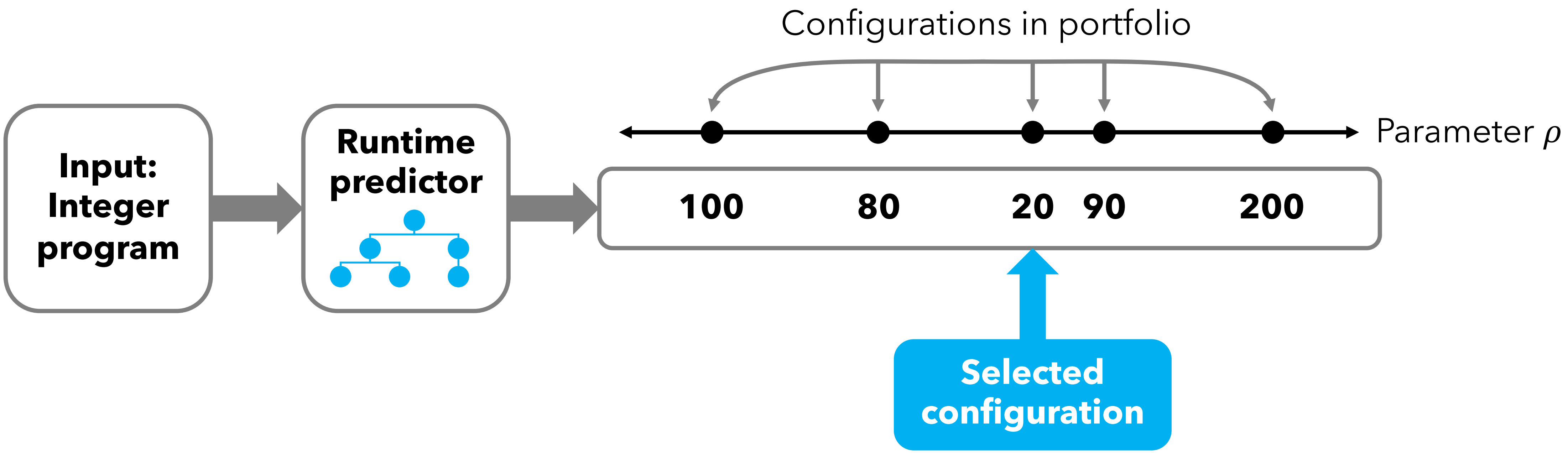
Algorithm portfolios

Best configuration for one problem is rarely optimal for another

Portfolio-based algorithm selection

1. Compile a diverse portfolio of parameter settings
2. At runtime, select one with strong predicted performance

Portfolio-based algorithm selection
Example



Example: integer programs

CombineNet: Platform for **sourcing auctions** (2001-2010)

Ran over 800 auctions, totaling over $60 billion 

These auctions require solving **large integer programs**

Used algorithm portfolios: **2-3x average speedup**

Sandholm [Handbook of Market Design '13]

Example: SATzilla

Algorithm portfolios used to sweep the 2007 SAT Competition

Xu, Hutter, Hoos, Leyton-Brown [JAIR'08]

Our contributions

First provable, end-to-end guarantees for using

machine learning in ❯ **portfolio-based algorithm selection**

Encompassing both:

1. Learning the **portfolio**
2. Learning the **algorithm selector**

Learning a portfolio & algorithm selector

1. Fix parameterized algorithm, e.g., CPLEX
2. Receive training set S of "typical" inputs, e.g., IPs

Problem instance 1 **Problem instance 2** **Problem instance 3** **Problem instance 4** ● ● ●

3. Use S to learn a **portfolio** $\hat{\mathcal{P}}$ of configurations and a **selector** $\hat{f}$ that maps problem instances to $\hat{\mathcal{P}}$

Learning a portfolio & algorithm selector

1. Fix parameterized algorithm
2. Receive training set S of "typical" inputs

Problem instance 1 | **Problem instance 2** | **Problem instance 3** | **Problem instance 4** | • • • | **Unseen?** **Problem instance**

3. Use S to learn a **portfolio** $\hat{\mathcal{P}}$ of configurations
   and a **selector** $\hat{f}$ that maps problem instances to $\hat{\mathcal{P}}$

Key question: On **future** inputs,
Will the configuration $\hat{f}$ selects have good performance?

Generalization error

Key question: On **future** inputs,
Will the configuration $\hat{f}$ selects have good performance?

Generalization error:
Difference between **avg** performance of $\hat{f}$ on training set
and **expected** (future) performance

Small generalization error ➡ **No overfitting**

Generalization error

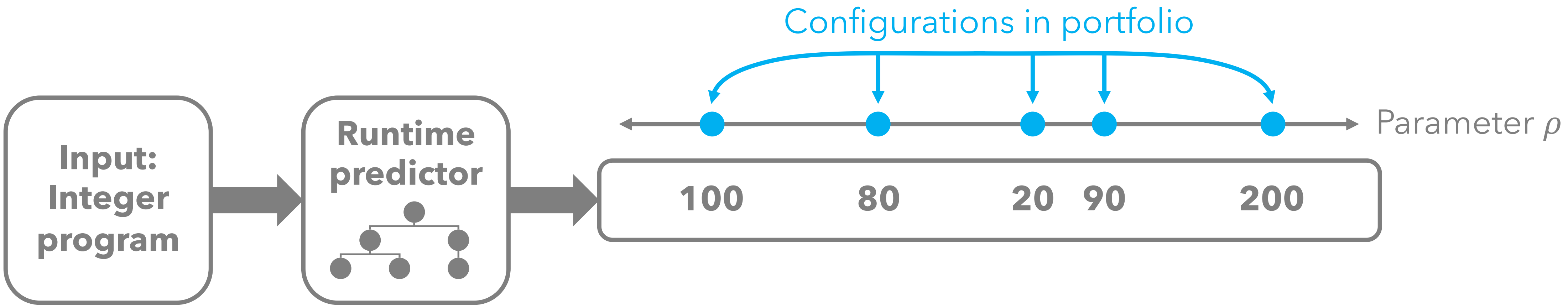
Key question: On **future** inputs,
Will the configuration $\hat{f}$ selects have good performance?

Generalization error:
Difference between **avg** performance of $\hat{f}$ on training set
and **expected** (future) performance

If we choose $\hat{\mathcal{P}}$, $\hat{f}$ to have good **average** performance,
we can also guarantee good **future** performance

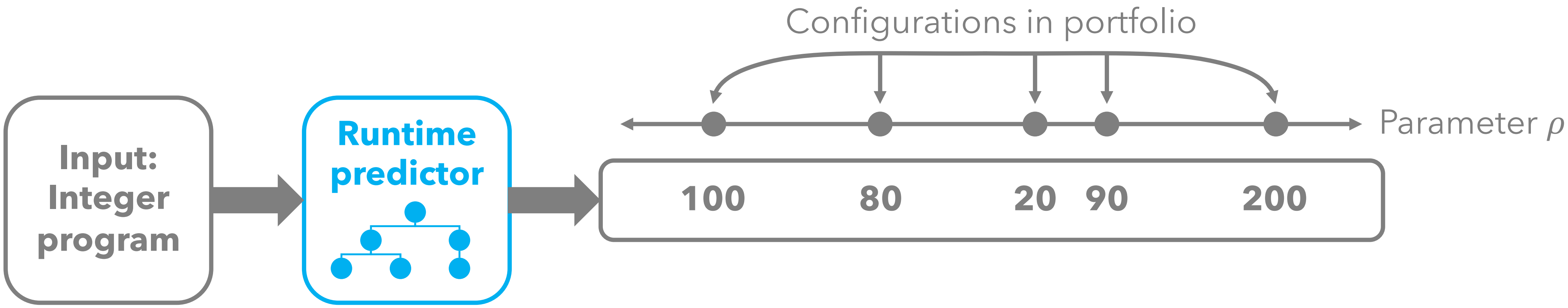
3 sources of generalization error

1) **Size** of the portfolio



3 sources of generalization error

1) **Size** of the portfolio
2) Learning-theoretic complexity of the **algorithm selector**



3 sources of generalization error

1) **Size** of the portfolio
2) Learning-theoretic complexity of the **algorithm selector**
3) Learning-theoretic complexity of:
   the algorithm's **performance** as a function of its parameters

Unlike prior work on algorithm configuration generalization e.g:

Gupta, Roughgarden	ITCS'16
Balcan, Dick, Sandholm, **Vitercik**	ICML'18
Garg, Kalai	NeurIPS'18

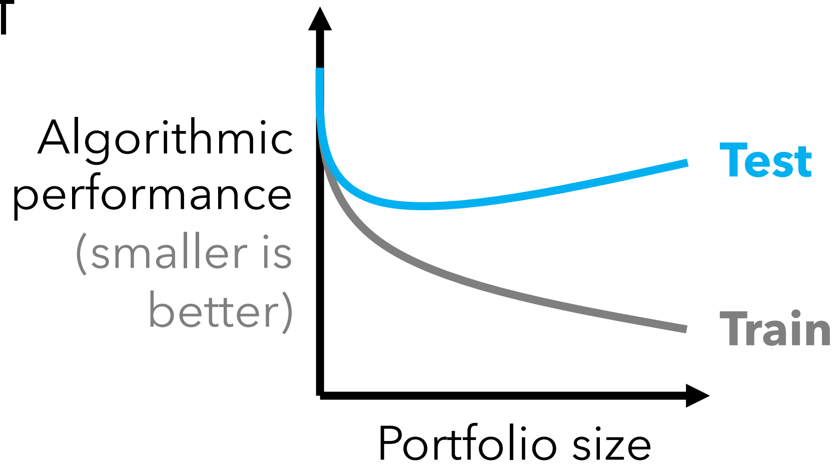
...which only had to contend with (3)

Our results: Main message

Our **theory** says:

As portfolio grows, can have good configuration for any input,
…but it becomes **impossible** to avoid **overfitting**

Our **experiments** illustrate this tradeoff



Outline

1. Introduction
2. **Model**
3. Main result
4. Implications for common algorithm selectors
5. Experiments
6. Conclusions and future directions

Model

$\mathcal{Z}$: Set of all inputs (e.g., integer programs)

$\mathbb{R}$: Set of all parameter settings (e.g., CPLEX parameter)

Standard assumption: Unknown distribution $\mathcal{D}$ over inputs

E.g., represents scheduling problem airline solves day-to-day

Algorithmic performance

$u_\rho(z)$ = utility of algorithm parameterized by $\rho \in \mathbb{R}$ on input z

E.g., runtime, solution quality, memory usage, …

Assume $u_\rho(z) \in [-1,1]$

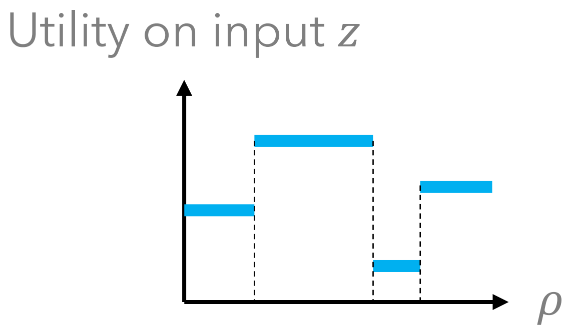
Can be generalized to $u_\rho(z) \in [-H, H]$

Algorithmic performance

$u_\rho(z)$ = utility of algorithm parameterized by $\rho \in \mathbb{R}$ on input z

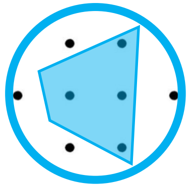
$u_z^*(\rho)$ = utility as a function of the parameter

Assumption: $u_z^*(\rho)$ is piecewise constant with $\leq t$ pieces



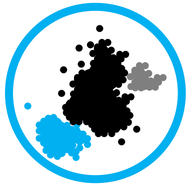
Algorithmic performance

Assumption: $u_z^*(\rho)$ is piecewise constant with $\leq t$ pieces



Integer programming

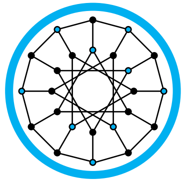
Balcan, Dick, Sandholm, **Vitercik**, ICML'18



Clustering

Balcan, Nagarajan, **Vitercik**, White, COLT'17

Balcan, Dick, White, NeurIPS'18; Balcan, Dick, Lang, ICLR'20



Greedy algorithms

Gupta, Roughgarden, ITCS'16

Computational biology

Balcan, DeBlasio, Dick, Kingsford, Sandholm, **Vitercik**, '20

Outline

1. Introduction
2. Model
3. **Main result**
4. Implications for common algorithm selectors
5. Experiments
6. Conclusions and future directions

Generalization error

Key question: On **future** inputs,
Will the configuration $\hat{f}$ selects have good performance?

Generalization error:
Difference between **avg** performance of $\hat{f}$ on training set
and **expected** (future) performance

Generalization error

Given **samples** $z_1, \ldots, z_N \sim \mathcal{D}$ and learned algorithm **selector** $\hat{f}$,

$$\Bigg|\underbrace{\frac{1}{N}\sum_{i=1}^{N} u_{\hat{f}(z_i)}(z_i)}_{\text{Average empirical utility of the configurations selected by } \hat{f}} - \underbrace{\mathbb{E}_{z\sim\mathcal{D}}\left[u_{\hat{f}(z)}(z)\right]}_{\text{Expected utility of the configuration selected by } \hat{f}}\Bigg| \leq \text{?}$$

Average empirical utility of the configurations selected by $\hat{f}$

Expected utility of the configuration selected by $\hat{f}$

Generalization error

Given **samples** $z_1, \ldots, z_N \sim \mathcal{D}$ and learned algorithm **selector** $\hat{f}$,

$$\left|\frac{1}{N}\sum_{i=1}^{N} u_{\underbrace{\hat{f}(z_i)}_{\text{Configuration selected by } \hat{f} \text{ given input } z_i}}(z_i) - \mathbb{E}_{z\sim\mathcal{D}}\left[u_{\hat{f}(z)}(z)\right]\right| \leq \textbf{?}$$

Generalization error

Given **samples** $z_1, \dots, z_N \sim \mathcal{D}$ and learned algorithm **selector** $\hat{f}$,

$$\left| \frac{1}{N} \sum_{i=1}^{N} \underbrace{u_{\hat{f}(z_i)}(z_i)}_{\text{Utility of the configuration selected by } \hat{f} \text{ given input } z_i} - \mathbb{E}_{z \sim \mathcal{D}}\left[u_{\hat{f}(z)}(z)\right] \right| \leq \textbf{?}$$

Generalization error

Given **samples** $z_1, \dots, z_N \sim \mathcal{D}$ and learned algorithm **selector** $\hat{f}$,

$$\left| \underbrace{\frac{1}{N}\sum_{i=1}^{N} u_{\hat{f}(z_i)}(z_i)} - \mathbb{E}_{z\sim\mathcal{D}}\left[u_{\hat{f}(z)}(z)\right] \right| \leq \textbf{?}$$

Average empirical
utility of the configurations
selected by $\hat{f}$

Generalization error

Given **samples** $z_1, \dots, z_N \sim \mathcal{D}$ and learned algorithm **selector** $\hat{f}$,

$$\left| \frac{1}{N} \sum_{i=1}^{N} u_{\hat{f}(z_i)}(z_i) - \underbrace{\mathbb{E}_{z \sim \mathcal{D}}\left[u_{\hat{f}(z)}(z)\right]} \right| \leq \mathbf{?}$$

Expected utility of the configuration selected by $\hat{f}$

Main result

With high probability over the draw $z_1, \ldots, z_N \sim \mathcal{D}$,

Intrinsic complexity of the set of algorithm selectors

Portfolio size

Number of pieces

$$\left| \frac{1}{N} \sum_{i=1}^{N} u_{\hat{f}(z_i)}(z_i) - \mathbb{E}_{z \sim \mathcal{D}}\left[u_{\hat{f}(z)}(z)\right] \right| = \tilde{O}\left(\sqrt{\frac{\boldsymbol{d} + \boldsymbol{\kappa} \log \boldsymbol{t}}{N}} \right)$$

Takeaway: No matter how we choose portfolio $\hat{\mathcal{P}}$ & selector $\hat{f}$,

Average performance is indicative of **future** performance

Main result

With high probability over the draw $z_1, \dots, z_N \sim \mathcal{D}$,

Intrinsic complexity of the set of algorithm selectors | Portfolio size | Number of pieces

$$\left|\frac{1}{N}\sum_{i=1}^{N} u_{\hat{f}(z_i)}(z_i) - \mathbb{E}_{z\sim\mathcal{D}}\left[u_{\hat{f}(z)}(z)\right]\right| = \tilde{O}\left(\sqrt{\frac{\boldsymbol{d} + \boldsymbol{\kappa}\log \boldsymbol{t}}{N}}\right)$$

Strong average performance ➡ Strong future performance

Main result

With high probability over the draw $z_1, \ldots, z_N \sim \mathcal{D}$,

$$\left|\frac{1}{N}\sum_{i=1}^{N} u_{\hat{f}(z_i)}(z_i) - \mathbb{E}_{z\sim\mathcal{D}}\left[u_{\hat{f}(z)}(z)\right]\right| = \tilde{O}\left(\sqrt{\frac{\underbrace{\boldsymbol{d}}_{\substack{\textit{Intrinsic complexity} \text{ of the} \\ \text{set of algorithm selectors}}} + \underbrace{\boldsymbol{\kappa}}_{\text{Portfolio size}} \log \underbrace{\boldsymbol{t}}_{\text{Number of pieces}}}{N}}\right)$$

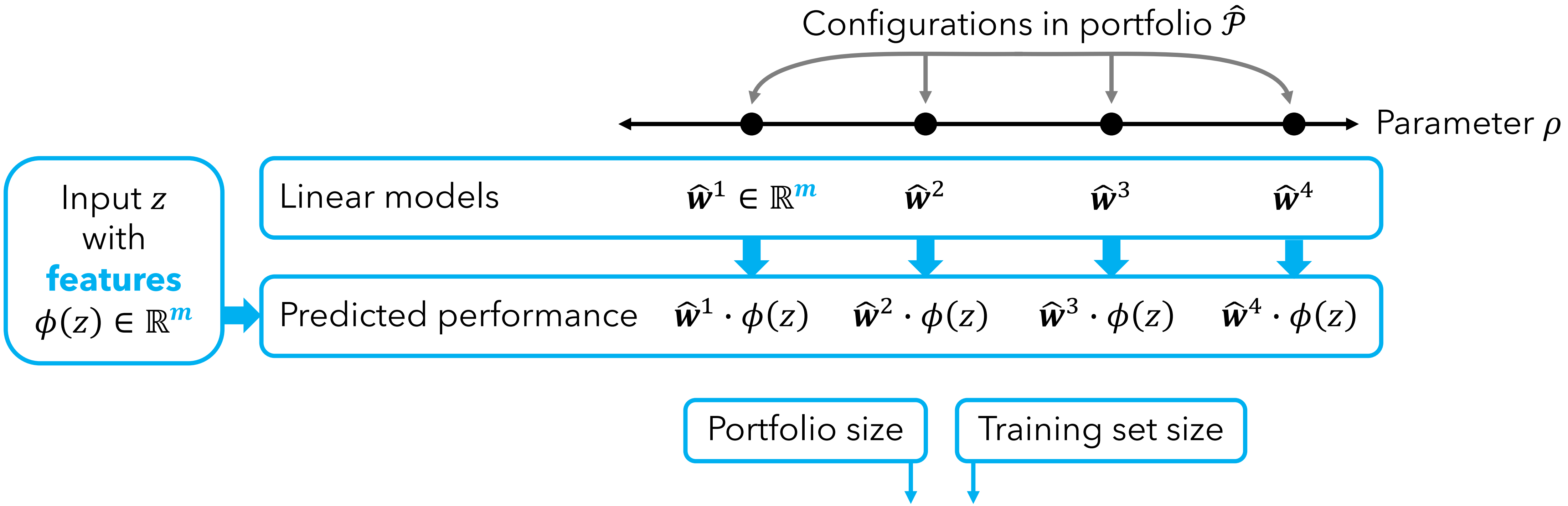
Nearly-matching **lower bound** of $\tilde{\Omega}\left(\sqrt{\frac{\boldsymbol{d}+\boldsymbol{\kappa}}{N}}\right)$

Outline

1. Introduction
2. Model
3. Main result
4. **Implications for common algorithm selectors**
5. Experiments
6. Conclusions and future directions

Linear performance models

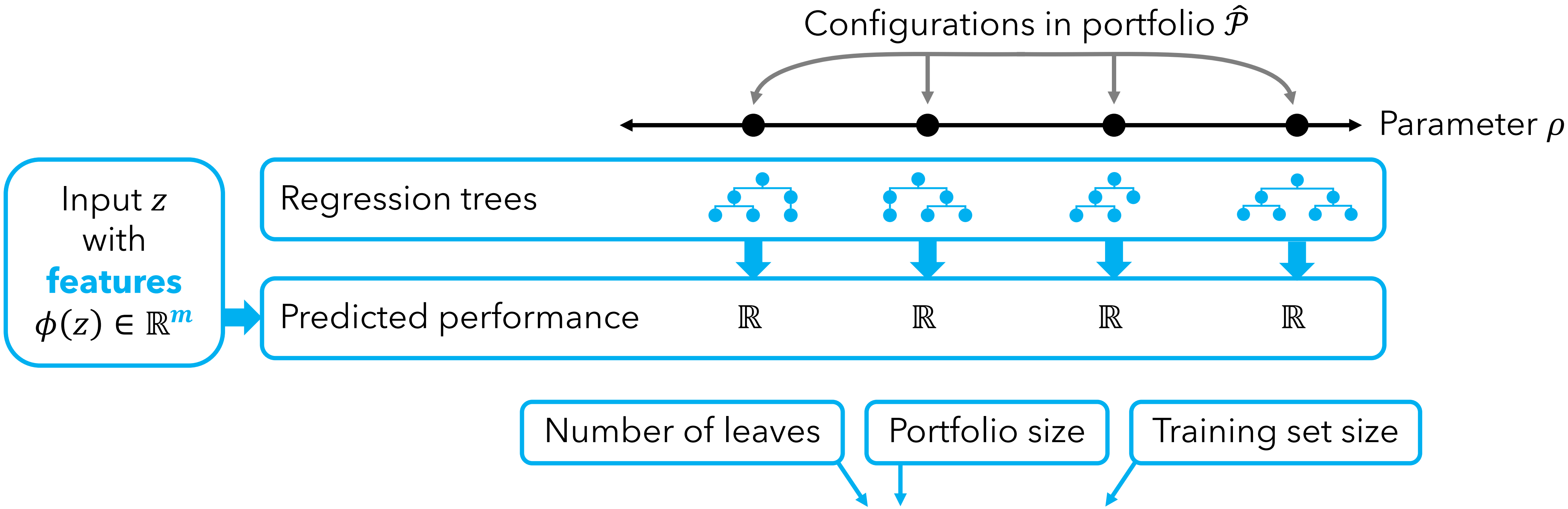
E.g., Xu, Hutter, Hoos, Leyton-Brown [JAIR'08]; Xu, Hoos, Leyton-Brown [AAAI'10]



Generalization error bound: $\tilde{O}\left(\sqrt{m\kappa/N}\right)$

Regression tree performance models

E.g., Hutter, Xu, Hoos, Leyton-Brown [AIJ'14]

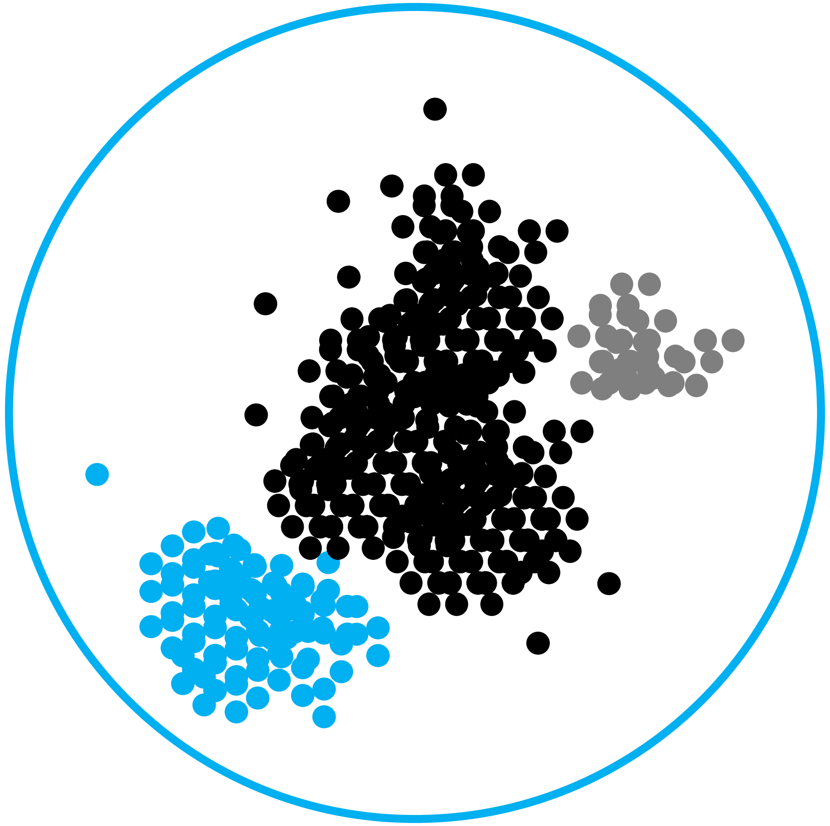


Generalization error bound: $\tilde{O}\left(\sqrt{\ell \kappa \log m / N}\right)$

Clustering-based algorithm selectors

Kadioglu, Malitsky, Sellmann, Tierney [ECAI'10]

See the paper!



Outline

1. Introduction
2. Model
3. Main result
4. Implications for common algorithm selectors
5. **Experiments**
6. Conclusions and future directions

Experiments: Integer programming

Branch and bound: Most widely-used IP algorithm

Used by commercial solvers such as CPLEX and Gurobi

Recursively partitions feasible region to find optimal solution

Organizes partition as a search tree

Experiments: Integer programming

Tune a **variable selection** policy parameter

Distribution over combinatorial auction IPs

Leyton-Brown, Pearson, Shoham [EC'00]

Portfolio selected greedily

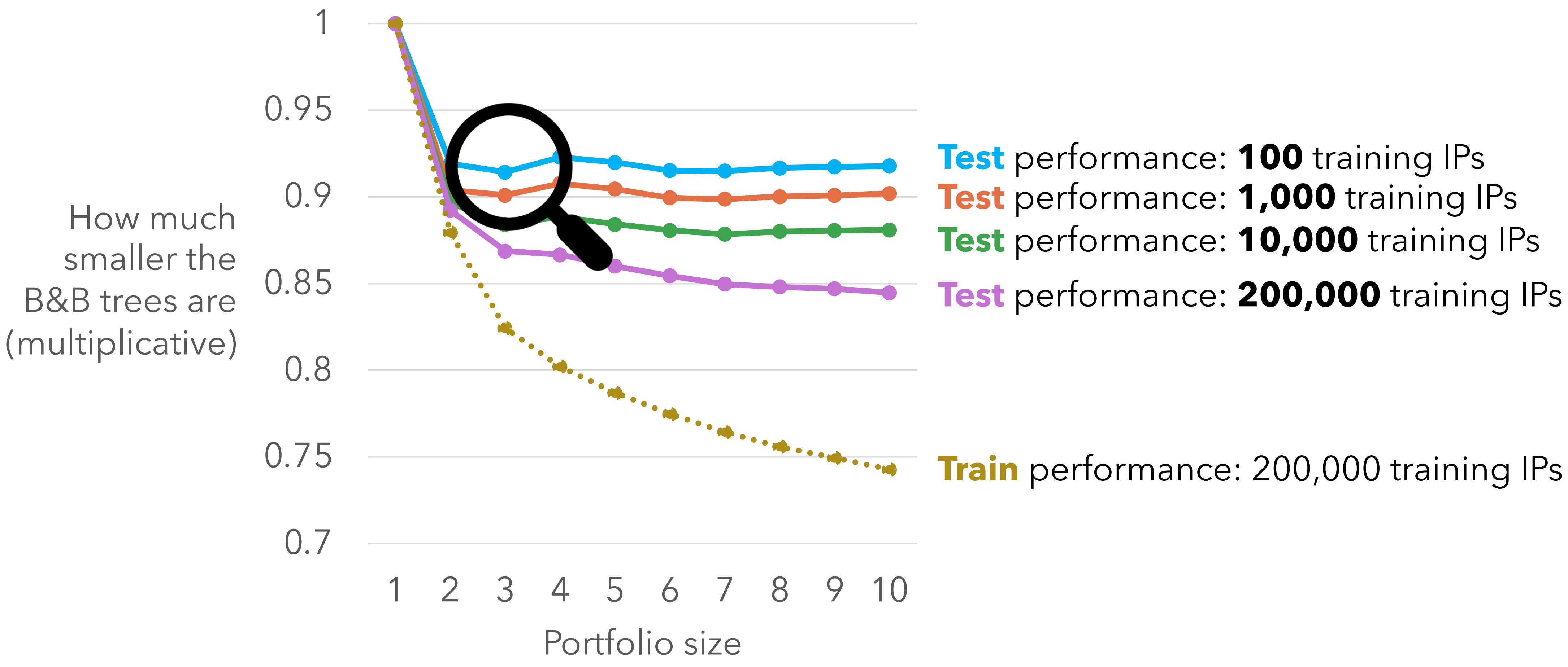
Regression forest performance model

Hutter, Xu, Hoos, Leyton-Brown [AIJ'14]

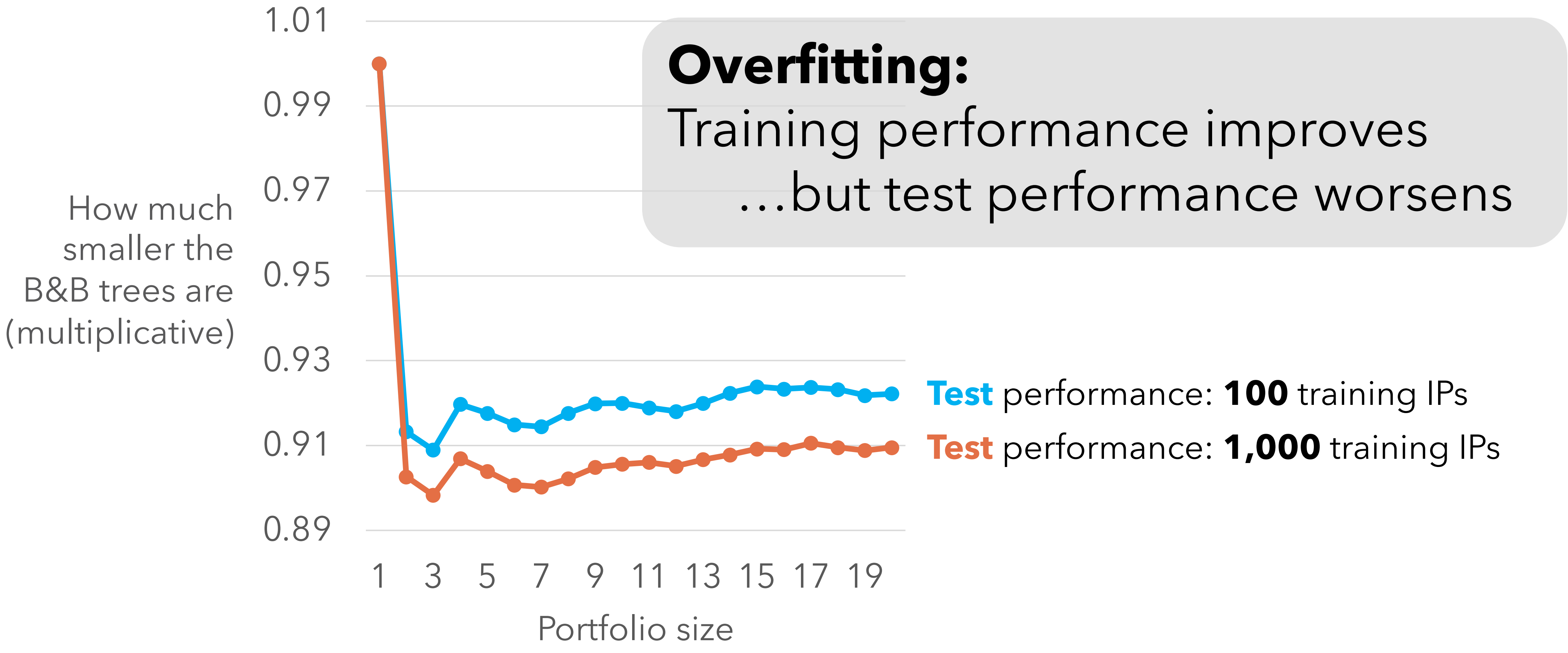
Features generated using open-source software

Leyton-Brown, Pearson, Shoham [EC'00]
Hutter, Xu, Hoos, Leyton-Brown [AIJ'14]

Experiments: Integer programming



Experiments: Integer programming



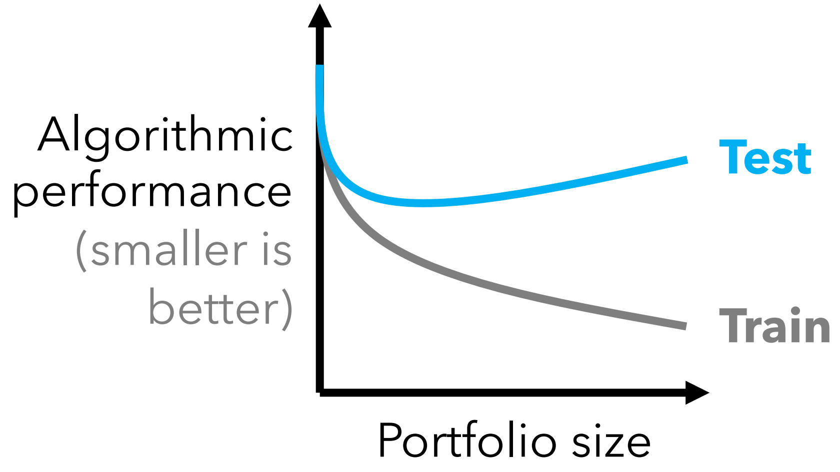
Outline

1. Introduction
2. Model
3. Main result
4. Implications for common algorithm selectors
5. Experiments
6. **Conclusions and future directions**

Conclusions and future directions

Theory and experiments illustrate a fundamental tradeoff:

As portfolio grows, can have good configuration for any input,
…but it becomes **impossible** to avoid **overfitting**



Conclusions and future directions

Theory and experiments illustrate a fundamental tradeoff:

As portfolio grows, can have good configuration for any input,
…but it becomes **impossible** to avoid **overfitting**

Future direction:
Does the **diversity** of a portfolio impact its generalization?